**Array Representation in Memory:**

Arrays are contiguous blocks of memory where each element is of the same data type. The array's starting point, known as the base address, is used to calculate the address of any element in the array. The address of the i-th element in an array can be computed using the formula:

Address(i)=Base Address+(i×Size of Each Element)\text{Address}(i) = \text{Base Address} + (i \times \text{Size of Each Element})Address(i)=Base Address+(i×Size of Each Element)

**Advantages of Arrays:**

1. **Direct Access:** Arrays provide O(1) time complexity for accessing elements by their index.
2. **Memory Efficiency:** Arrays have no extra overhead for storing metadata, which makes them memory efficient.
3. **Cache Friendly:** Due to their contiguous memory allocation, arrays are cache-friendly and offer better performance in terms of access speed.

**Limitations:**

1. **Fixed Size:** The size of the array is fixed at the time of creation and cannot be changed. This can lead to wasted memory if the array is too large or out-of-bounds errors if the array is too small.
2. **Costly Insertions and Deletions:** Insertions and deletions (other than at the end) require shifting elements, which can be costly (O(n) time complexity).
3. **Inefficient Search:** Searching for an element in an unsorted array takes O(n) time.

**When to Use Arrays:**

1. **Fixed Number of Elements:** When the number of elements is known and will not change.
2. **Direct Access Needed:** When fast direct access to elements by index is required.
3. **Memory Efficiency:** When memory efficiency is a priority, and the array size can be predetermined.